Challenge 6

CY6740 – Machine Learning in CyberSecurity

Name: Tejas Krishna Reddy

NUID: 001423166

Solution:

Preprocessing: Load the dataset which is in the form of Attribute Relationship File Format (ARFF) using the scipy.io.arff module. While you load you can observe that there are 97019 records available in the dataset in which each of the record has 26 features in it as described in the problem statement.

Part 1:

- Using Isolation Forest algorithm we could find:

```
# Print the number of samples that have been classified as -1 (outliers)
# Returns 1 for inliers and -1 for outliers
print("Num of outliers detected by Isolation Forest Algorithm = ",list(clf.predict(dataset)).count(-1))
```

```
Num of outliers detected by Isolation Forest Algorithm =  12079
```

Part 2:

```
### Part 2: Decision Function
dfun = clf.decision_function(dataset)
```

```
### Average of all anamoly scores in the dataset:
print("Average anomaly score from Isolation forest algorithm on dataset = ",np.mean(dfun))
```

```
Average anomaly score from Isolation forest algorithm on dataset =  0.08354828681590722
```

-

```
# Values less than -0.2:
print("Number of instances that have less then -0.2 anamoly score = ", len(np.where(dfun <= -0.2)[0]))
```

```
Number of instances that have less then -0.2 anamoly score =  7
```

```
# Rows that have anamoly score less than -0.2 are
print('Index of Rows that have anamoly score less than -0.2 are:')
np.where(dfun <= -0.2)[0]
```

```
Index of Rows that have anamoly score less than -0.2 are:
]: array([73134, 73819, 74597, 75647, 78490, 79224, 80418], dtype=int64)
```

Part 3:

Local Outlier Factor Algorithm:

```
# Define LOF
clf1 = LocalOutlierFactor(n_neighbors=3)
```

```
# Num of instances labeled as outliers
print("Num of instances labeled as outliers = ", list(clf1.fit_predict(dataset)).count(-1))
```

```
Num of instances labeled as outliers =  10564
```

Part 4:

```
# Calculating average LOF scores for all the instances
clf1_scores = clf1.negative_outlier_factor_
print("Average LOF scores of all instances = ", np.mean(clf1_scores))
```

```
Average LOF scores of all instances =  -2.6230392330832236e+33
```

```
# Number of instances that have less or equal to -40 LOF score
print("Number of instances that have LOF less than or equal to -40 = ", len(np.where(clf1_scores <= -40)[0]))
```

```
Number of instances that have LOF less than or equal to -40 =  302
```

The code for the above results has been attached below for your reference:

Challenge 6 - Outlier and Anomaly Detection Techniques

- **Author: Tejas Krishna Reddy**
- **NUID: 001423166**

Necessery python modules to be installed-

- !pip install pandas
- !pip install numpy
- !pip install scipy
- !pip install sklearn

In [1]:
```
import pandas as pd
import numpy as np
from scipy.io import arff
from sklearn.ensemble import IsolationForest
from sklearn.neighbors import LocalOutlierFactor
```

Load the ARFF Data:

In [2]:
```
data, meta = arff.loadarff('DatasetChallenge6.arff')
dataset = np.array(data.tolist(), dtype=np.float64)
# Print an example of how the data (26 features) of each instance looks like
dataset[1]
```

Out[2]:
```
array([4.00000000e+00, 4.00000000e+00, 1.83000000e+02, 2.33000000e+02,
       9.00000000e+00, 1.80000000e+01, 3.00000000e+00, 1.00000000e+01,
       3.00000000e+00, 1.00000000e+01, 0.00000000e+00, 4.10000000e+01,
       1.90000000e+01, 1.15000000e+02, 2.00000000e-01, 5.00000000e-01,
       1.00000000e+00, 0.00000000e+00, 2.00000000e+01, 2.00000000e+00,
       1.00000000e+00, 0.00000000e+00, 0.00000000e+00, 0.00000000e+00,
       5.40229857e-01, 1.04309767e+00])
```

Isolation Forest

In [3]:
```
### Part 1: Isolation Algorithm
clf = IsolationForest(random_state=0).fit(dataset)
```

In [5]:
```
# Print the number of samples that have been classified as -1 (outliers)
# Returns 1 for inliers and -1 for outliers
print("Num of outliers detected by Isolation Forest Algorithm = ",list(clf.
```

```
Num of outliers detected by Isolation Forest Algorithm =  12079
```

In [6]:
```
### Part 2: Decision Function
dfun = clf.decision_function(dataset)
```

In [7]:
```
### Average of all anamoly scores in the dataset:
print("Average anomaly score from Isolation forest algorithm on dataset = "
```

```
Average anomaly score from Isolation forest algorithm on dataset =  0.08354
828681590722
```

In [25]:
```
#Values less than -0.2:
print("Number of instances that have less than -0.2 anomoly score = ", len(
```

```
Number of instances that have less than -0.2 anomoly score =  7
```

In [16]:
```
# Rows that have anamoly score less than -0.2 are
print('Index of Rows that have anamoly score less than -0.2 are:')
np.where(dfun <= -0.2)[0]
```

```
Index of Rows that have anamoly score less than -0.2 are:
```

Out[16]:
```
array([73134, 73819, 74597, 75647, 78490, 79224, 80418], dtype=int64)
```

Local Outlier Factor Algorithm:

In [17]:
```
# Define LOF
clf1 = LocalOutlierFactor(n_neighbors=3)
```

In [18]:
```
# Num of instances labeled as outliers
print("Num of instances labeled as outliers = ", list(clf1.fit_predict(data
```

```
Num of instances labeled as outliers =  10564
```

In [19]:
```
# Calculating average LOF scores for all the instances
clf1_scores = clf1.negative_outlier_factor_
print("Average LOF scores of all instances = ", np.mean(clf1_scores))
```

```
Average LOF scores of all instances =  -2.6230392330832236e+33
```

In [22]:
```
# Number of instances that have less or equal to -40 LOF score
print("Number of instances that have LOF less than or equal to -40 = ", len
```

```
Number of instances that have LOF less than or equal to -40 =  302
```

In [23]:
```
# All index of instances that have score <= -40:
print("All index of instances that have LOF less than or equal to -40:")
np.where(clf1_scores <= -40)[0]
```

```
All index of instances that have LOF less than or equal to -40:
```

Out[23]:
```
array([ 7421, 15861, 15868, 16205, 16210, 16302, 16309, 16326, 16333,
       22031, 28030, 38221, 38232, 38233, 39680, 40911, 41225, 41399,
       41402, 41410, 41417, 41679, 42586, 46147, 48390, 50264, 58993,
       59711, 63055, 63057, 63059, 63062, 63064, 63066, 63249, 63252,
       63254, 63256, 63259, 63261, 63263, 63266, 63268, 63271, 63273,
       63278, 63280, 63282, 63287, 63294, 63301, 63484, 64334, 65941,
       66152, 66156, 66160, 66165, 66169, 66173, 66178, 66182, 66186,
       66191, 66195, 66199, 66204, 66208, 66212, 66221, 66225, 66230,
       66234, 66238, 66243, 66247, 66251, 66256, 66260, 66264, 66269,
       66273, 66277, 66281, 66286, 66290, 66294, 66299, 66303, 66307,
       66312, 66316, 66320, 66325, 66329, 66333, 66338, 66342, 66346,
       66351, 66381, 66385, 66390, 66394, 66398, 66403, 66411, 66416,
       66420, 66424, 68390, 68394, 68398, 68403, 68407, 68411, 68554,
       68777, 68781, 68786, 68790, 68794, 68799, 68803, 68808, 68813,
       68817, 68822, 68826, 68831, 68835, 68840, 68849, 68854, 68858,
       68863, 68868, 68872, 68877, 68881, 68886, 68891, 68895, 68900,
       68904, 68909, 68913, 68917, 68922, 68927, 68931, 68936, 68940,
       68945, 68949, 68954, 68959, 68963, 68968, 68972, 68977, 68982,
       68986, 70386, 70391, 70395, 70399, 70404, 70408, 70551, 71953,
       72027, 72036, 72286, 73134, 73293, 73386, 73512, 73569, 74371,
       74619, 75310, 75583, 75825, 76250, 76374, 76909, 77095, 77675,
       77758, 78786, 79119, 79156, 79228, 79475, 79545, 79563, 79605,
       79732, 81219, 81615, 81764, 81790, 81904, 82332, 82812, 82885,
       82887, 82904, 83102, 83453, 83544, 83765, 83767, 83777, 83786,
       83809, 83840, 83849, 83859, 83861, 83866, 83873, 83875, 83898,
       83903, 83905, 83910, 83912, 83959, 83982, 84038, 84052, 84083,
       84097, 84116, 84128, 84130, 84142, 84151, 84165, 84174, 84186,
       84214, 84219, 84221, 84228, 84278, 84308, 84317, 84324, 84340,
       84345, 84353, 84365, 84372, 84374, 84393, 84395, 84407, 84409,
       84419, 84421, 84442, 84454, 84461, 84470, 84477, 84489, 84500,
       84505, 84519, 84535, 84540, 84542, 84547, 84557, 84587, 84603,
       84615, 84629, 84643, 84654, 84670, 84677, 84712, 84721, 84735,
       84760, 84764, 84776, 84813, 84824, 84832, 84842, 84851, 84853,
       84878, 84885, 84900, 85490, 85491], dtype=int64)
```

In [ ]: